REMARKS

This responds to the Office Action mailed on March 8, 2004.

This response cancels no claims, amends claims 1, 33-34, 38-39, 43-44, 46-49, and 52, and adds new claims 55-67. As a result, claims 1-5, 7-9, 11, and 31-67 remain pending in this Application.

Claims 34 and 52 were objected to due to informalities. The amendments made herein to the claims obviate the objections, and do not affect their scope.

Claims 1-5, 7-9, 11 and 31-54, all of the claims currently pending, were rejected under 35 USC §102(b) as anticipated by Asghar et al. (U.S. 5,794,068). Applicant respectfully traverses these rejections.

Applicant's embodiments physically separate "essential" code that actually executes an application program from "nonessential" code generated to speed up the execution of that application program, and that need not---and sometimes should not---be executed when the application runs. Although the two code streams are stored separately, they are both executed in the same hardware microarchitecture structure. The motivation for this separate storage but common execution is that the nonessential help code may actually hinder the essential code if it were intermixed in the same stream and executed unconditionally along with it.

The embodiments taught by Asghar are configured differently and serve a different purpose. Asghar allows an application program written for a single general-purpose execution unit (GP core) to run faster on a system having an additional processor (DSP core) that can execute a subset of the instructions more efficiently. A function preprocessor converts certain program instruction types into different instructions that execute only on the specialized execution unit.

Amended claim 1 expresses these differences. If Applicant's "first pipeline"[1] for "essential code" is read on Asghar's stream of program instructions, then the only possible equivalent of the "second pipeline" would be Asghar's DSP instructions that exist only in his dual execution unit system. However, Asghar then does not meet the claim recitation to a

[1] --- The amendments to the recitations for the first and second pipelines merely provide explicit antecedents for "instructions" later in the claim. They do not change its meaning or affect its scope.

"single" microarchitecture structure that executes "instructions both from the essential code and from the nonessential code." That is, Asghar's GP core executes only those instructions in the processor's general-purpose instruction set, and his DSP core executes only those specialized instructions in its own DSP instruction set, which differs from the general-purpose set. In fact, these two instruction sets apparently have no instructions in common, no instructions that could be executed by both cores.

Paragraph 13 of the Office Action asserts in discussing claim 33 that Asghar's GP core and DSP core could be considered a "single microarchitecture structure" because they physically reside in the same CPU. This argument is specious. The term "microarchitecture" is commonly understood to refer to the hardware (and possibly microcode) that execute an instruction set at the lowest level in a digital processor, the level immediately above the digital logic or gate level. See, for example, Tannenbaum, STRUCTURED COMPUTER ORGANIZATION, 4th Ed., 1999 (Prentice-Hall, Inc.), pp. 4-6 and 203-205, copy attached hereto. The passage cited in par. 13 of the Office Action, col. 4 lines 11-67, in fact extols the dual nature of Asghar's two different cores. "Since the DSP core is optimized for these DSP-type mathematical operations, the DSP core can generally execute the desired function in a reduced number of instructions and clock cycles." (col. 4 lines 42-46). Further, Asghar maintains that having two execution units instead of a single one is advantageous in that "[t]he DSP core also operates in parallel with the general purpose core, providing further performance benefits" (col. 4 lines 65-67. Therefore, Asghar cannot be said to have a "single" execution unit that corresponds to Applicant's "single microarchitecture structure."

Claim 1 is also amended to substitute the broader term "memory" for the term "pipeline." "Pipeline" can have a broad meaning, and was applied in this way to the reference, since Asghar does not teach a hardware multistage pipeline in the narrower sense. However, the term "memory" states explicitly that Applicant intends the broader sense in this context.

New dependent claim 55 carries claim 1 further by reciting that the single microarchitecture structure is able to execute "all" instructions from "both the essential and the nonessential code." That is, this claim covers cases such as H-flow code (e.g., Specification, page 4 line 26 to page 5 line 2) where the instructions used in the essential program code and the instructions used in the nonessential hint code are indistinguishable from each other. In Asghar, again, they are entirely distinct. New claim 56 covers cases such as virtualization (see, e.g., Specification page 16 lines 25-30) where the second pipeline may also contain some "code that is not nonessential." The DSP instructions in Asghar's function decoder are all nonessential, in that the program will execute correctly even if only the original GP instructions are run, and the DSP instructions are deleted.

Previously presented claims 2-5, 7-9, 11, and 31-32 also depend from claim 1, and thus incorporate all its recitations. In addition, claim 3, for example, couples the second pipeline to a cache of instructions that provide "hints for the execution of the instructions" in the first pipeline. The Office Action incorrectly equates "hints" with Asghar's operands for DSP instructions." Hints are well known in the art as items that "increase efficiency of operation" of program code (Specification page. 3 lines 26-37, e.g.), and are not necessary to correct operation of the program code. Operands, however, are necessary for correct operation, and have nothing to do with efficiency. As to claim 4, Applicant is unable to find anything in Asghar's register file (454, Fig. 4), his abstract, or the cited passage in cols. 3-4 that mentions a state of his processor cores for any purpose, much less for a "conjugate mapping table." The Examiner is respectfully requested to specifically identify a particular item with such a state. As another example, claim 9 recites a trigger comprising "a vector value"—that is, where multiple conditions initiate the nonessential code. Asghar appears to employ only single conditions: the detection of a particular GP instruction(s) triggers corresponding DSP instruction(s). The Examiner is requested to point out specifically in what way Asghar may utilize multiple conditions. As to claim 31, Applicant respectfully requests an interpretation of how Asghar's recognition of certain instructions in the program code can be said to "generate non-essential code" (that is, code that does not actually execute a program function). Applicant also finds absolutely no mention in the cited passages or in any other place of the "directed acyclic graph" recited in claim 32.

Method claim 33 amends the "loading" recitations to provide an explicit antecedent for "instructions" later in the claim, and substitutes the more accurate term "executing" for "processing." These amendments make the claim clearer, but do not affect its scope.

If the first "loading" operation of the claim is matched to loading Asghar's stream of program instructions, the second "loading" operation could correspond, if at all, only to Asghar's

DSP instructions that exist only in his dual execution unit system. However, Asghar then does not meet the claim recitation of executing instructions from the nonessential code "in the same microarchitecture structure" as the one that executes the essential code. As explained in connection with claim 1, Asghar's GP core executes only those instructions in the processor's general-purpose instruction set, and his DSP core executes only those specialized instructions in its own different DSP instruction set, and neither core executes any instructions designed for the other core. The meaning of the term "microarchitecture structure" is also delineated in connection with claim 1.

Dependent claims 34-48 distinguish the cited reference for the same reasons as does parent claim 33, and for other reasons as well. For example, claim 33 states that the first and second memories are "pipelines." In this context,[2] a pipeline is a multistage hardware device. Storing the two different code streams in two different pipelines allows execution to switch between them very quickly. This becomes a distinct advantage where the purpose of one of the streams is to accelerate the execution of the other stream. Asghar has no showing of a pipeline, or of any particular kind of memory. Claim 35 again names a particular type of memory, a "cache" in which items are stored for reuse and replaced from another memory upon a cache miss. Executing both the essential code and the nonessential code from caches allows, e.g., more individually tailored least-recently-used replacement of the cache contents. Although Asghar might (or might not) execute his program code from a cache, col. 4 lines 29-41 indicates that the DSP instruction code is statically stored for indexing when required, and is not replaced as in a cache.

Amended claim 38 recites a particular type of hint code, the typical hint code that is "generated by a compiler from the essential code," as described in the Specification at page 1 lines 29 to page 2 line 2. and at page 19 lines 25-26. Asghar's DSP code cannot be fairly characterized as "hint code" in the context of the present Application. In any case, however, Asghar's DSP sequences are fixed for all programs, and are generated for the particular DSP hardware, and not for individual programs at compile time.[3]

[2] --- That is, claim 35 names another particular memory structure.
[3] --- Claim 31 delineates another way to produce hint code, dynamically by a code analyzer.

Dependent Markush claims 39, 43, and 44 are expanded to separate their constituent elements. Claim 39 employs code the sequences for "instruction set virtualization." Par. 19 of the Office Action argues that Asghar that the DSP performs this function. However, Asghar actually performs an inverse of virtualization. As described in the Application, virtualization posits a stream of (essential code, in Applicant's system) instructions that are not within the architecture of a given processor, and replaces them (from Applicant's "nonessential" code) by one or more equivalent instructions that can be executed by that same architecture. In Asghar, the instructions for one architecture (the GP core) are translated for execution on a different architecture (the DSP core). New claims 57-61 list the other items from former claim 39. Asghar has no suggestion of performing "interrupt or exception processing" (claim 57), "speculative execution" (58), "security checking or sandboxing" (59), "test[ing] the microarchitecture" (60), or "prefetch[ing] essential code" into a memory (61). Former claim 43 is expanded into claims 43(amended) and 62-64. As above in connection with claim 39, Asghar cannot be said to "virtualize" either "instructions" (claim 39), or "blocks of instructions (62). Asghar suggests no virtualization of "registers" (63) or other "processor hardware resources" (64) in either of his cores. Claims 44(amended) and 65-67 delineate trigger types. As to triggers comprising "data attributes" (65), "state attributes" (66) within the processor, or "event attributes" (67), Applicant finds nothing in the passages cited in pars. 26-28 of the Office action for anything except instructions, and respectfully requests an explanation as to how this might apply to the particular items of claims 65-67 and 46-48.[4]

Independent claim 49 has been amended without changing its scope to explicitly include matters that were inherent in the previous version, and to rearrange its format for more logical presentation. Hint code may be produced by a compiler in the course of producing object code for an application program. That is, the hint code is personalized to the particular program, and need not be generic to all programs. Claim 49 embodies this feature in its declaration that the second part of the library includes nonessential code "associated with the same particular program." Asghar, on the other hand, produces only generic code that odes not vary from one program to the next. The DSP instruction sequences are permanently stored in his function decoder, and are applied in the same way to any program that happens through his processor.

[4] --- In view of the amendment to claim 44, claims 46-48 are changed to depend from claims 65-67 respectively.

The DSP code sequences cannot not differ from program to program, nor can the conditions under which they are applied.

As in claim 1, claim 49 substitutes the term "memory" for "pipeline," in order to clarify that Applicant intends any form of digital storage for these elements. The memory containing the library is renamed "storage," merely in order to avoid confusion; "storage" and "memory" are equivalent in this setting.

Dependent claims incorporate the features of parent claim 49 and others as well. For instance, claim 52 specifies that essential and nonessential code are stored "in separate sections of a file." Applicant finds nothing in the cited passages of Asghar that mentions files in any way. In fact, Asghar generates the DSP instructions, operands, and parameters that he substitutes for GP instructions in his function preprocessor (col. 4 lines 27-37), a unit that does not store application code. As another example, claim 52 avers that both the essential and the nonessential code reside "in a static file," as noted on page 19 lines 8-16 ""Static binary"). Applicant finds no mention of any static file in Asghar---in fact Asghar never combines both these types of code into any single file.

Conclusion

For the above and other reasons, Applicant urges that the claims meet all statutory requirements, and respectfully requests reexamination and allowance thereof. The Examiner is invited to telephone Applicant's attorney at (612) 373-6971 to facilitate prosecution of this Application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

Respectfully submitted,

HONG WANG ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
Attorneys for Intel Corporation
P.O. Box 2938
Minneapolis, Minnesota 55402
(612) 373-6971

Date 9 Aug 2004

By [illegible]
J. Michael Anglin
Reg. No. 24,916

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop ~~AF~~ RCE (KL 8/9/04), Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 9 day of August, 2004.

KACIA LEE
Name

Kacia Lee
Signature